
1.	1
1.	Luca Calibration (luca)	1
1.1.	Shuffled Complex Evolution (SCE)	2
1.2.	Luca calibration script	3
1.2.1.	Step - parameter sub element	7
1.2.2.	Step - objfunc sub-element	9
1.2.3.	ObjFunc Subelements sim, obs, and subdivide	11
1.3.	Luca examples	13
1.3.1.	PRMS model, 3 parameters, strategy MEAN, timestep DAILY	13
1.3.2.	PRMS model, 1 parameter, strategy INDIVIDUAL, timestep MEAN_MONTHLY	13
1.3.3.	PRMS model, 1 parameter, strategy BINARY, timestep ANNUAL_MEAN, period_range 2-7	14
1.3.4.	Monthly Water Balance model, 2 parameters, timestep MONTHLY_MEAN,	14
1.3.5.	Calibration selection example	15
1.4.	References	15
Index	17

Chapter 1.

1. Luca Calibration (_{luca})

Luca (Let us calibrate) is a multiple-objective, stepwise, automated procedure for model calibration (Hay and Umemoto, 2006). It was originally developed by the United States Geological Survey (USGS) for use with the PRMS model compiled with the USGS's Modular Modeling System (MMS) (Leavesley and others, 1996). The USGS version of Luca was a wizard-style user-friendly graphical user interface (GUI) that provides a systematic way of building and executing a calibration procedure. The calibration procedure uses the Shuffled Complex Evolution global search algorithm (Duan and others, 1994) to calibrate user selected model parameters.

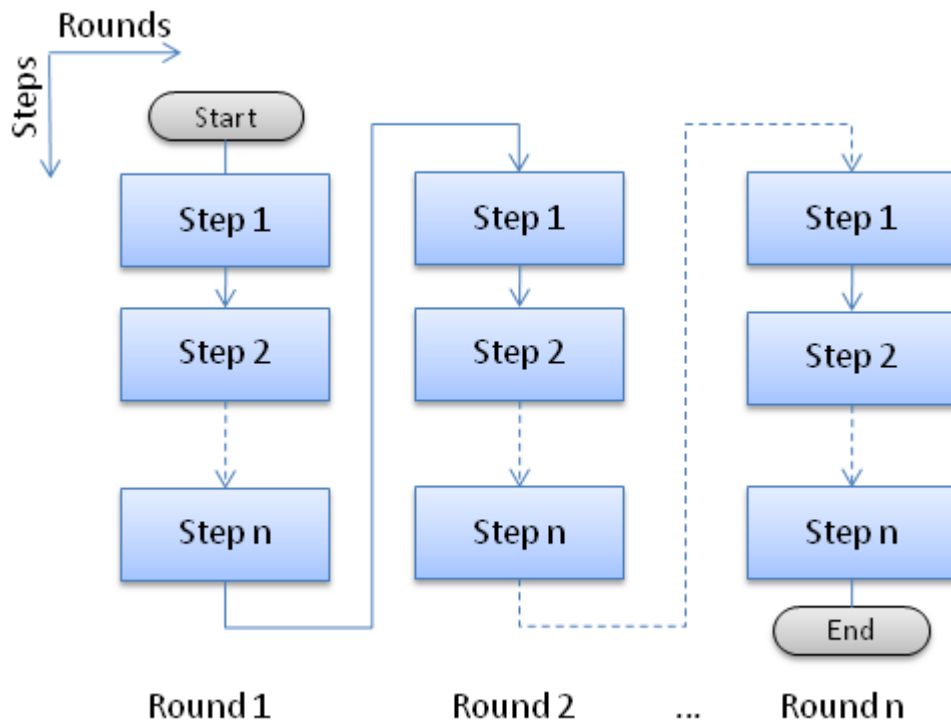
The Object Modeling System (OMS) (David and others, 2002, 2010; <http://www.javaforge.com/project/oms>) project has revised the Luca interface for applications within OMS. The revision changes the user interface for Luca from a wizard-style GUI to a script-based interface that is compatible with applications of OMS using the OMSSConsole. The OMS version of Luca provides all the calibration functionality of the original USGS version. The OMS version also includes the added ability to calibrate selected parameters on sub basins within a larger watershed, and adds the option to use custom objective functions, and works with any model that is integrated into OMS.

Where appropriate, sections of the original Luca users manual are used in this document to insure consistency in descriptions, definitions, and terminology so as to minimize confusion between documents and among users of both the MMS and OMS frameworks. In the use of quotes from the original Luca users manual, you can substitute OMS for MMS in almost all instances to describe the application of Luca in OMS.

The Luca framework is based on the concepts of steps and rounds. A step is composed of a user-defined set of parameters to be calibrated using one or more user-selected objective functions. A round is the sequential computation of all defined steps. The original Luca users manual describes the relation of steps and rounds well.

“Luca requires a user-defined number of steps, which are executed sequentially within a user-defined number of rounds. To start the calibration procedure, an initial parameter file containing all MMS parameters is defined. The parameters identified for each calibration step are calibrated. These calibrated parameter values replace the respective parameter values in the parameter file, and this parameter file is used as the initial parameter file for the next calibration step. Completion of the user-designated number of steps constitutes a round. Once a parameter is calibrated, its value is set for the remainder of that calibration round. This process is repeated until the user-designated number of steps and rounds are completed.”

Figure 1.1. Rounds and Steps in Luca



As noted, the primary change in the use of Luca in OMS is the use of a Luca script, denoted by the suffix of .luca appended to the file name, to define the steps and rounds, and the control elements of the the Luca computational procedures. The details of the .luca script and selected examples are provided below.

1.1. Shuffled Complex Evolution (SCE)

The Luca calibration procedure uses the Shuffled Complex Evolution (SCE) methodology. It is described in the original users manual as follows.

Duan and others (1994) provide a detailed description of the SCE algorithm implemented in Luca. The SCE global optimization algorithm, developed by Duan and others (1992), addresses the difficulties in optimization when there are several regions of attraction and multiple local optima in the parameter space. SCE avoids the problem of being trapped in local optima by using a population-evolution-based global search technique, which searches for the optimal solutions from a population of possible solution points, rather than a single point. ...

In SCE, the set of parameters to be calibrated is considered a point in N-dimensional space, where N is the number of parameters to be optimized. SCE randomly samples S points in the feasible parameter space. The MMS executable is run with each point (parameter set) and an objective function is calculated. The objective function determines how close the simulation results are to “observed” values. The S objective function values are sorted by increasing order (where lowest is the “best” fit) and then divided into a user-defined number of complexes (P), each containing M points. Each complex is evolved by using the Competitive Complex Evolution algorithm (based on the Nelder and Mead (1965) Simplex Downhill search algorithm). The points in the evolved complexes are combined into a single-sample population. The sample population is sorted by increasing objective function value and shuffled into P complexes.

The shuffling loop is created and repeated until the results of the complex evolution meet one of the following convergence criteria: (1) the number of MMS executions reaches the maximum number of model executions ...; (2) the percentage change in the best objective function value of the current shuffling loop and that of several shuffling loops before is less than a specified percentage ...; or (3) the points converge into a very small region, which is less than 0.1 percent

of the space within the lower and upper bounds of parameters. With each consecutive shuffling loop, the number of complexes created decreases by one ($P=P-1$). This decrease stops when the number of complexes reaches the minimum number of complexes required (P_{min}). The output is the parameter file containing the point (a parameter set) that has the best objective function value.

The steps in the SCE procedure as described above can be summarized as:

1. **Generating points.** The set of parameters to be calibrated is considered as a point in N dimension space where N is the number of parameters. SCE generates many points, in which each parameter has a random value within its lower and upper bound values.
2. **Assigning criterion values.** The model is run with every point (a set of parameters) generated in SCE Step 1 as an input. An objective function that determines how close the simulation results are to observed values is used to calculate a criterion value for each point.
3. **Creating complexes.** The points are divided into smaller groups called complexes such that points of good and bad criterion values are equally distributed.
4. **Complex evolution.** Each complex is evolved in the following way: Several points are selected from the complex to construct a sub-complex. In the sub-complex, a new point is generated, and a point that has a bad criterion value is replaced with this new point. This evolution step is repeated several times with different random points in a sub-complex.
5. **Combining complexes.** All points in the complexes are combined together to be one group.
6. **SCE Steps (3) – (5) are called a shuffling loop.** It is repeated until the results of the complex evolution meet one of the following end conditions:
 - The number of model executions reaches the maximum number of model execution
 - The percent change in the best criterion value of the current shuffling loop and that of several shuffling loops before is less than a specified percentage.
 - The points converge into a very small region, which is less than 0.1% of the space within the lower and upper bounds of parameters.

The number of complexes used in SCE Step 3 decreases by 1 for every shuffling loop. This decrease stops when the number of complexes reaches the **minimum number of complex required**. The output is the parameter file containing the point (a parameter set) that has the best criterion value.

1.2. Luca calibration script

A Luca calibration is executed using a OMS script file has the extension `.luca`. It is similar to the OMS simulation script file with a `.sim` extension. The `.luca` file provides information on the model, parameter file and data file to be used in the calibration procedure. In addition, it provides information on the initialization period and calibration period, and the sets of parameters and objective functions that will be used in the calibration process. For purposes of this discussion, the features and functions of the components of the `luca` file will be described in terms of these basic script components.

The calibration sequence is defined as a series of steps and rounds. A step is associated with a selection of parameters from a given input-parameter file. A round consists of one or more steps. The calibration proceeds one step at a time. At the completion of a step, the calibrated values of the parameters selected in that step are written to the working parameter file and passed into the next step. This sequence of passing the working parameter file to the next step is repeated until all steps are executed. The sequence of calibration steps is then repeated for the designated number of rounds.

An example luca file is shown below. It has 2 rounds and 2 steps.

```
/*  
* Luca calibration
```

```

*/
import static oms3.SimBuilder.instance as OMS3

OMS3.luca(name: "EFC-luca") {

    // define output strategy: output base dir and
    // the strategy NUMBERED|SIMPLE|DATE
    outputstrategy(dir: "$work/output", scheme:NUMBERED)

    // define model
    model(classname:"model.PrmsDdJh") {
        // parameter
        parameter (file:"$work/data/efcarson/efc_test2_params.csv") {
            inputFile   "$work/data/efcarson/data.csv"
            outFile      "out.csv"
            sumFile      "basinsum.csv"
            out           "summary.txt"

            startTime    "1980-10-01"
            endTime      "1985-09-30"
        }
    }

    output(vars:"date, basin_cfs, runoff[0]", fformat="7.3f", file:"out1.csv")

    summary_file "efc_test2_summary.txt"

    calibration_start "1982-10-01"           // Calibration start date
    start_month_of_year 10
    rounds 2                                // calibration rounds, default 1

    // step definitions
    // step 1
    step {
        parameter {
            jh_coef(lower:0.0005, upper:0.09, calib_strategy:INDIVIDUAL, subset:"0-*")
        }

        objfunc(method:ABSDIF, timestep:MEAN_MONTHLY, period_range:"1-12") {
            sim(file:"out1.csv", table:"EFC-luca", column:"basin_potet")
            obs(file:"$oms_prj/data/efcarson/efcrs_PEObs_monthOnly.csv",
                table:"obs",column:"PE")
        }
    }

    // step 2
    step {
        parameter {
            soil2gw_max(lower:0.05, upper:0.5, calib_strategy:MEAN)
            ssrcoef_sq(lower:0.01, upper:0.5, calib_strategy:MEAN)
            tmax_allsnow(lower:30.0, upper:36.0, calib_strategy:MEAN)
        }

        objfunc(method:ABSDIF, timestep:DAILY, period_range:"1-12") {
            sim(file:"out1.csv", table:"EFC-luca", column:"basin_cfs")
            obs(file:"$oms_prj/data/efcarson/data_lucatest.csv", table:"obs",
                column:"runoff[0]")
        }
    }
}

```

The first section of the .luca file provides the standard information OMS needs to execute a model. It begins with a user-defined name of the Luca calibration. In this example the name is EFC-test2. This name will be used to name a subdirectory created in the output directory of the OMS project. The subdirectory will contain the results of the calibration process. The name of the .luca file is its single property but it has a number of Elements related to the property.

luca Properties

Name	Description	Type	Required
name	the name of the calibration	String	y

The elements of the .luca file are shown below.

luca Elements

Name	Description	Type	Default	Occurrences
outputstrategy{}	output management		StandardOutput	?
model{}	the model to execute			?
output	simulation resource definition	String		*
summary_file	Specifies the name of a summary file to create	String	no file generated if no name specified	1
trace_file	Specifies the name of a calibration trace file to create	String	no file generated if no name specifies	1
calibration_start	start date of calibration	ISO Date String	-	1
start_month_of_year	The first month of the 12-month year. Used to define the annual period for ObjFunc calculations.	int : 1 (Jan) - 12 (Dec)	1 (Jan)	1
rounds	number of rounds	int	1	?
step{}	calibration step definition		-	+

The **outputstrategy{}** element defines the type of output files that will be generated by the .luca run. The options are

- **NUMBERED** - a new subdirectory is created for each run and is sequentially numbered starting with 0000
- **SIMPLE**- only one subdirectory is created and it is overwritten for each run
- **DATE** - a new subdirectory is created for each run and is sequentially named with the date and time

The **model{}** element provides the name of the model to be calibrated and the associated information needed to run the model. This information includes the path and name of the parameter file(s), the path and name of the data file, the names of any other input or output files associated with the model, and the *startTime* and *endTime* values for the model run.

The *startTime* and *endTime* values define the total period of the model run. The total period is composed of an initialization period that runs the model through one or more wetting and drying cycles, and then a calibration period where the selected objective functions are computed for parameter calibration purposes. The initialization period runs from the *startTime* to the day prior to the user-defined **calibration_start** element. The calibration period then runs from *calibration_start* to the *endTime*. The initialization period is intended to minimize any initial biases in the estimated starting parameter values.

The **output** element declares the model variables that are to be written to a specified output file. These variables are those that are to be used in the objective function computations in all the steps of the Luca calibration. In this example *basin_cfs* and *runoff[0]* are the simulated and observed daily streamflow respectively.

The **summary_file** element specifies the name of the file that will contain the summary information related to the Luca calibration results. Information in the summary file includes the initial and calibrated parameter values and objective function values.

The **trace_file** element specifies the name of the csv file that will contain information about the objective function and parameter values for each iteration of the calibration algorithm. The results are broken up by round and step.

The **start_month_of_year** specifies which month is the start of the annual (12 month) period for calculation purposes. In the United States, water balance and other hydrologic measures are often computed on a water year basis where the water year starts in the month of October (10). Other regions of the world may use the calendar year which starts in the month of January (1). The start_month_of_year allows users to specify the month that is appropriate for their purpose.

The **rounds** element defines the number of times that all the calibration steps will be executed. The **step** element is where the details of each calibration step are provided. The step has a number of sub elements which are shown in the table below. The step has a single property which is a name. If no name is provided, it will be given a number.

The sub elements **parameter** and **objfunc** are required to be provided by the user. The remaining sub elements are parameters that relate to the steps in the SCE computational process as described above in section 1.1. Each of the SCE parameters has a default value computed for it in each step based on the number and type of parameters selected in the step. The user can override one of more of these default values, but they should be familiar with the computational mechanics of SCE to insure the integrity of the SCE process. If no SCE parameter values are provided, the default values will be used.

step Properties

Name	Description	Type	Required
name	the name of the step	String	no

step Sub elements

Name	Description	Type	Default	Occurrences
parameter{ }	parameter to calibrate	-	-	+
objfunc{ }	objective function definition	-	-	+
max_exec	maximum # executions in one step	int	10000	?
init_complexes	Initial number of complexes	int	2	?
points_per_complex	Number of points in each complex	int	2*(# param values)+1	?
points_per_subcomplex	Number of points in a sub-complex	int	(# param values) + 1	?
evolutions	Number of evolution steps before shuffling	int	2*(# param values) + 1	?
min_complexes	Minimum number of complexes required	int	1	?
shuffling_loops	Shuffling loops in which the objective function value must	int	5	?

Name	Description	Type	Default	Occurrences
	change by given % before optimization is terminated			
of_percentage	Percentage for the objective function value (Range: 0-1)	double	0.01	?

1.2.1. Step - parameter sub element

The **parameter** sub element can contain one or more parameters that are to be calibrated in the step. For each parameter, a set of properties are available to specify details of the calibration procedure for each selected parameter. A simple example with one parameter is shown below.

```
parameter {
  jh_coef(lower:0.0005, upper:0.09, calib_strategy:MEAN, subset"0-*", filter_param:"hru_subbasin")
}
```

The lower and upper values are the lower and upper bounds on the parameter to be calibrated.

Name	Description	Type	Required
lower	the lower boundary	double	y
upper	the upper boundary	double	y
calib_strategy	the calibration strategy	MEAN INDIVIDUAL BINARY	n (default:MEAN)
filter_param	The name of the filter parameter for calibration selection	string	n (default: match subset to parameter value index instead of this filterParam value)
subset	selects which subset of params are to be used for calibration	String ("m-n,p,...")	n. (default: include all parameter's values for calibration)
subset_col	Used as the column subset for 2D array tables	String	n
subset_row	Used as the row subset for 2D array tables	String	n

The **calib_strategy** has one of three possible approaches that can be selected.

- **MEAN**: Optimize based on the mean value of the parameter instances.
- **INDIVIDUAL**: Optimize each parameter instance independently.
- **BINARY**: Treat each individual parameter instance as a binary value. Optimize each individually.

filter_param and **subset** can be used to select which of the spatially or temporally distributed parameters are to be used in calibration.

The user can specify neither of these, subset only, or both.

- If neither of these are included, then all parameter values will be included in calibration.

- If only subset is specified without filter_param, then only parameter values with indexes matching any value in the subset will be included in calibration.
- If both filter_param and subset are specified, then parameter values for which the corresponding filter_param value is equal to any value in the subset will be included in calibration
- For 2D parameters, use subset_col and subset_row in place of subset in order to specify the desired columns and rows. If subset is specified, it will act as subset_col. subset selection for 2D arrays using a filter_param is untested, and so is not supported.

Examples:

Use all jh_coef_hru values for calibration:

```
jh_coef_hru(lower:1, upper:100, calib_strategy:MEAN)
```

Use jh_coef values 0-5 for calibration, where e.g. these might correspond to jh_coef monthly values for Jan-June:

```
jh_coef(lower:1, upper:100, calib_strategy:MEAN, subset:"0-5")
```

Use all jh_coef_hru values with hru_subbasin=1 (i.e. use only the jh_coef_hru values for hru's within subbasin 1)

```
jh_coef_hru(lower:1, upper:100, calib_strategy:MEAN, filter_param:"hru_subbasin", subset:"1" )
```

Use subset_col and subset_row for a 2D parameter

```
jh_coef_hru(lower:1, upper:100, calib_strategy:MEAN, subset_col:"3-4", subset_row:"1,3,5" )
```

Note that calibration selection with filter_param and subset can be used with any calib_strategy.

MEAN

The details of mean were described in the original Luca manual as:

When the **mean** is chosen, the mean parameter value (instead of each individual parameter value) is calibrated. Each time SCE generates a value for the mean, individual parameter values are generated based on the new mean such that the mean-value distribution is preserved. This option is a good choice when a spatially distributed parameter is chosen for calibration.

When the mean value is chosen as the calibration type, the individual parameter values must be regenerated from the SCE-generated mean value. Given n-individual initial parameter values (P_{init_n}), the new individual parameter values (P_n) are reproduced from the SCE-generated mean ($MEAN_{SCE}$) by using the following equation:

$$P_n = \{[(MEAN_{SCE} + C) * (P_{init_n} + C)] / \{MEAN_{INIT} + C\}\} - C$$

where $MEAN_{INIT}$ is the mean of n-initial parameters (P_{init_n}). C is a constant used to avoid zero values in equation (1):

$$C = [\text{absolute value of the user defined lower bound}] + 10.0$$

The user must designate a Lower Bound and an Upper Bound for each selected parameter. The lower bound cannot be greater than the minimum parameter value and the upper bound cannot be less than the maximum parameter value. These bounds are used to guide the generation of points in SCE. In SCE, a parameter set is considered a point in N-dimensional space, where N is the number of parameter values in the parameter set. The initial individual values or mean values of the parameter set displayed in Instruction 4-2 are used as one of the points in SCE. The rest of the points are randomly generated by SCE such that each parameter value is within its lower and upper bounds.

After entering the values for lower and upper bounds, the Actual Range sampled for the mean in SCE is displayed when Use the mean value is chosen (red box in fig. 6). This range is calculated based on the user-defined lower and upper bounds such that no individual values are out of

range. Actual Range refers to the sampling range used in SCE. The Actual Range is calculated as follows:

INDIVIDUAL

When **INDIVIDUAL** is chosen, each individual value of a distributed parameter is calibrated. All or a subset of individual values of a parameter can be selected using the range element. A caution from the original users manual is

The user is cautioned against using this choice when a parameter is dimensioned by more than 12. After entering the values for lower and upper bounds, the actual range sampled for the individual values in SCE is displayed. The actual range is equal to the user-defined bound when Use the individual values is selected as the calibration type.

The **INDIVIDUAL** option is typically used when there are external measures available against which to compare model computed results for each individual parameter value. An example of this is the calibration of the monthly parameter in the Jensen-Haise potential evapotranspiration (PET) equation. The user can specify an external file with measured or estimated mean monthly PET that can be used with the simulated mean monthly PET for each month to compute the selected objective function. See the example of this in the Examples section of this document.

BINARY

The **BINARY** option is used only for those parameters that consist of values of 0 or 1. An example might be a model that uses 0 and 1 to specify a subset of temperature or precipitation stations to use in the computation of distributed temperature and precipitation in a basin. All or a subset of individual values of a binary parameter can be selected using the range element. The description used in the original users manual is

When Parameters are binary (0,1) is chosen for calibration ..., the parameter values for the highlighted parameter must be either 0 or 1... The Lower Bound and Upper Bound are (0,1) for this choice. When the SCE-calibrated value is greater than or equal to 0.5, the parameter value is set to 1. Otherwise, it is set to 0. If all parameter values are chosen for calibration, then keep in mind that they may all be set to 1 (or 0) at any time in the SCE process. The user should be familiar with their model. If the model requires one of the binary parameters to be set to 1 (or 0), the user must ensure that the ... Initial Parameter Value are set accordingly.

1.2.2. Step - `objfunc` sub-element

The objective function sub-element **objfunc** specifies the mathematical form of the objective function, the time period over which the objective function is computed, and the simulated (sim) and observed (obs) variables to be used in the objective function, and the path to the variable locations. Luca also supports the use of multiple objective functions for a step. The **objfunc** sub-element properties as shown in the table below.

Name	Description	Type	Required
method	the objective function	See objective function table below	
timestep	the time step for simulated and observed values	RAW TIME_STEP DAILY MONTHLY_MEAN MEAN_MONTHLY TANNUAL_MEAN PERIOD_MEDIAN PERIOD_MIN PERIOD_MAX PERIOD_STDEV	n (default: DAILY)
weight	the objective function weight	double (0 - 1.0)	n ²⁾
period_range	Defines which months to include in analysis	String ("m-n,p,...") where m,n,p are 1-12 for Jan-Dec	n (default "1-12")

Name	Description	Type	Required
subdivide_value	If a subdivide file is specified, only include the dates matching these values in the subdivide file when computing the ObjFunc	String("m-n,p,...")	n (must be specified if a subdivide file is used.)

The **method** defines the mathematical function that will be used to compute the objective function. The methods available and their names are shown in the table below.

Table 1.1. Objective Function methods

Name	Description	Equation
ABSDIF	Absolute difference	$\sum_{i=1}^n Q_{i,o} - Q_{i,s}$
ABSDIFLOG	Absolute difference Log	$\sum_{i=1}^n \ln Q_{i,o} - \ln Q_{i,s} $
AVE	Absolute Volume Error	$\sum_{i=1}^n Q_{i,s} - Q_{i,o} $
IOA	Index of Agreement	$1 - \frac{\sum_{i=1}^n Q_{i,o} - Q_{i,s} }{\sum_{i=1}^n Q_{i,s} - Q_o + Q_{i,o} - Q_o }$
IOA2	Index of Agreement (Pow 2)	$1 - \frac{\sum_{i=1}^n (Q_{i,o} - Q_{i,s})^2}{\sum_{i=1}^n (Q_{i,s} - Q_o)^2 + (Q_{i,o} - Q_o)^2}$
NS	Nash-Sutcliffe	$1 - \frac{\sum_{i=1}^n (Q_{i,o} - Q_{i,s})^2}{\sum_{i=1}^n (Q_{i,o} - Q_o)^2}$
NSLOG	Log of Nash-Sutcliffe	$1 - \frac{\sum_{i=1}^n \ln Q_{i,o} - \ln Q_{i,s} }{\sum_{i=1}^n \ln Q_{i,o} - \ln Q_o }$
NS2LOG	Log of Nash-Sutcliffe (Pow 2)	$1 - \frac{\sum_{i=1}^n (\ln Q_{i,o} - \ln Q_{i,s})^2}{\sum_{i=1}^n (\ln Q_{i,o} - \ln Q_o)^2}$
BIAS	BIAS	$\frac{\sum_{i=1}^n (Q_{i,o} - Q_{i,s})}{\sum_{i=1}^n Q_{i,o}}$
PMCC	Pearson product-moment correlation coefficient	$\frac{\sum_{i=1}^n (X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_{i=1}^n (X_i - \bar{X})^2} \sqrt{\sum_{i=1}^n (Y_i - \bar{Y})^2}}$
RMSE	Root Mean Square Error	$\sqrt{\frac{1}{n} \sum_{i=1}^n (Q_s - Q_o)^2}$
TRMSE	Transformed Root Mean Square Error	$\sqrt{\frac{1}{n} \sum_{i=1}^n (Z_s - Z_o)^2}$, $Z = \frac{(1+Q)^{0.3} - 1}{0.3}$

The **timestep** defines how the user-selected simulated and observed variable will be used to compute the objective function (OF). The available timesteps are

- **RAW**: take data exactly as-is without applying the period-range or assuming any timescale. Timestamp on data is completely ignored. It is up to the user to ensure that the time of the Observed and Simulated data match line for line.

- **TIME_STEP** : take data as-is but apply the period-range, without assuming any timescale. It is up to the user to ensure that the time of the Observed and Simulated data match line for line.
- **DAILY** : a value for each day.
- **MEAN_MONTHLY** : the arithmetic mean of the monthly means of a given month during a specified period of years. For example, the mean monthly value for January is the mean of all January mean values computed over the period of years used in the OF computation.
- **MONTHLY_MEAN** : the arithmetic mean of values for a given month. For example, the monthly mean is computed as the sum of the means of all months in the period specified, divided by the total number of months in the period.
- **ANNUAL_MEAN** : the mean of all data within each year. The annual period (i.e. year) starts in start_month_of_year.
- **PERIOD_MEDIAN** : the median of all data within the period_range of months each year. The annual period (i.e. year) starts in start_month_of_year
- **PERIOD_MIN** : the minimum of all data within the period_range of months each year. The annual period (i.e. year) starts in start_month_of_year
- **PERIOD_MAX** : use the maximum of all data within the period_range of months each year. The annual period (i.e. year) starts in start_month_of_year

The **weight** is used when multiple objective functions are defined for a step. The user can assign a different weight to each objective function. The sum of the weights assigned must equal 1.0. If no weights are assigned by the user, equal weight will be given to each objective function.

The **period_range** allows the user to specify a period of months for the objective function computation. If the period selected is 1-5, then the objective function will be computed for the months of January through May. If the start_month_of_year is October (10) and the selected period of objective function computation is October through February, the period_range can be specified as (10-12, 1-2) or (1-2, 10-12). The order of the sequence of months is not critical.

A **subdivide_value** is used when an external file is provided to specify specific times when the associated objective function is to be computed. For example, if a user wants to calibrate the parameters in this step using only days below a specified streamflow rate (low flow period), then an external file can be prepared that contains a date and an integer value for each day within the data range. For example, in the external file, the value 1 could be assigned to specify all days that are below the user-selected streamflow value. All other days would be assigned an integer value different from 1. The objective function then will only be computed on days that the value in the external file equals the subdivide_value, in this example case a value of 1.

The name and location of the simulated (sim) and observed (obs) variables to be used in the computation of the objective function are defined using the sim, obs, and, in some cases, the subdivide properties of the objfunc subelement. These objfunc properties are shown in the table below.

1.2.3. ObjFunc Subelements sim, obs, and subdivide

The objfunc element has 3 subelements, describing the data to be used to compute the objective function value from.

Name	Description	Type	Default	Occurrences
sim	the simulated variable information	sim{ }	-	1
obs	the observed variable information	obs{ }	-	1
subdivide	subdivide file containing information	subdivide{ }	if not specified, do not use subdivide file	?

Name	Description	Type	Default	Occurrences
	about which dates to use this objfunc for.			

Each of these properties is specified using the *file*, *table*, and *column* attributes of a standard OMS file structure. The *file* attribute is the path to the specified file containing the variable. The *table* is the name of the table that contains the variable, and the *column* is the column name within this table that contains the variable. The example below shows a file generated by the **output** element of Luca. The file name will be the name assigned in the **output** element and the *table* name is the name of the Luca script. The @T indicates the start of table information, with the *table* name EFC-luca being the first element. The @H indicates the header information which provides the names of each column. The date column is followed by four columns of daily output variables that were specified in the **output** element of the Luca script (described above).

```
@T, "EFC-luca"
  created_at, "Tue Feb 21 16:05:35 MST 2012"
  date_format, yyyy-MM-dd hh:mm:ss
@H, date, basin_cfs, basin_potet, runoff[0], swrad[0]
  type, Date, Double, Double, Double, Double
, 1980-10-01 12:00:00, 116.453, 0.189, 84.000, 568.171
, 1980-10-02 12:00:00, 117.684, 0.176, 82.000, 562.129
, 1980-10-03 12:00:00, 113.514, 0.187, 80.000, 560.905
, 1980-10-04 12:00:00, 112.072, 0.177, 80.000, 554.866
, 1980-10-05 12:00:00, 110.649, 0.159, 80.000, 546.476
```

An example of the **objfunc** sub-element is shown below. In this example the sim variable is located in the file generated by the **output** element of the Luca script and so the path to this file is assumed to be in the OMS project output directory. However, the obs variable is being taken from the input data file being read by the model and the path to this file is defined relative to the top level of the OMS project directory which is written as \$oms_prj. The path to the subdivide file is also specified using the \$oms_prj convention.

```
objfunc(method:ABSDIF, timestep:DAILY, period_range:"1-12", subdivide_value:1) {
  sim(file:"out1.csv", table:"EFC-luca", column:"basin_cfs")
  obs(file:"$oms_prj/data/efcarson/data_lucatest.csv", table:"obs", column:"runoff[0]")
  subdivide(file:"$oms_prj/data/efcarson/efc_subdivide.csv", table:"sd", column:"sd_data")
}
```

The subdivide file in this example was created using an external program that assigned the integer value of 1 to all days with streamflow below a low flow threshold value. The subdivide file format, shown below, uses the standard OMS data file structure. The table (@T) and header (@H) fields provide information about the table contents. The *column* name *sd_data* is the *column* that is identified in the **objfunc** sub-element. The *table* name and header *column* names are the only required elements of the subdivide table, but other information about the source of the data and the streamflow station is also permitted but not required.

```
@T,sd,
  created_at,"May 29, 2008",
  created_by,george,
  converted_from,efcarson.data,
  date_start,1980 10 1 0 0 0,
  date_end," ""1986 09 30 0 0 0""",
  date_format,yyyy MM dd H m s,
,,
@H,date,sd_data
name,,E FK CARSON
ID,,10308200
elevation,,5400
x,,-119.7648985
y,,38.7146274
  type,Date,Real
,1980 10 1 0 0 0,1
,1980 10 2 0 0 0,1
,1980 10 3 0 0 0,1
```

```
,1980 10 4 0 0 0,1
,1980 10 5 0 0 0,1
```

1.3. Luca examples

1.3.1. PRMS model, 3 parameters, strategy MEAN, timestep DAILY

This step example has 3 parameters from the PRMS model. The first two parameters are distributed parameters. The soil2gw_max has a value for each hydrologic response unit (HRU), the ssrcoef_sq has a value for each sub-surface reservoir, and tmax_allsnow is a scalar parameter. The strategy MEAN is used for all parameters. In this case the relative magnitudes of the parameter values in each of the distributed parameters will be maintained.

The objective function method selected is the absolute value of the difference between the daily values of the simulated and observed values of streamflow. Here basin_cfs is the simulated value and obs is the observed value.

```
step {
  parameter {
    soil2gw_max(lower:0.05, upper:0.5, calib_strategy:MEAN)
    ssrcoef_sq(lower:0.01, upper:0.5, calib_strategy:MEAN)
    tmax_allsnow(lower:30.0, upper:36.0, calib_strategy:MEAN)
  }

  objfunc(method:ABSDIF, timestep:DAILY) {
    sim(file:"out1.csv", table:"EFC-test1-luca", column:"basin_cfs")
    obs(file:"$work/data/efcarson/data_lucatest.csv", table:"obs", column:"runoff[0]")
  }
}
```

1.3.2. PRMS model, 1 parameter, strategy INDIVIDUAL, timestep MEAN_MONTHLY

This step example has a single parameter from the PRMS model and uses the INDIVIDUAL strategy for calibration. The parameter, jh_coef is a set of monthly coefficients used in the Jensen-Haise potential evapotranspiration (PET) computation. The range of 0-* indicates that all 12 values of the jh_coef will be used in the objective function computation.

```
step {
  parameter {
    jh_coef(lower:0.0005, upper:0.09, calib_strategy:MEAN)
  }

  objfunc(method:ABSDIF, timestep:MEAN_MONTHLY, period_range:"1-12") {
    sim(file:"out1.csv", table:"EFC-test-luca", column:"basin_potet")
    obs(file:"$oms_prj/data/efcarson/efcrs_PEObs_monthOnly.csv",
        table:"obs", column:"PE")
  }
}
```

The objective function method selected is the absolute value of the difference between the mean monthly values of the simulated and estimated PET. The mean monthly simulated PET is computed for each month by summing the daily values for that month and dividing by the number of days used in the sum. The result is the mean daily value for that month. The observed, or estimated observed, value for each month is provided using an external table of values, one value for each of the 12 months. The table uses the standard OMS format. The external table used in this example is shown below.

```
@T,obs
created_at,9/29/2011
created_by kmolson,
converted_from,efcarson.data
date_format,"MM"
```

```
@H,date,PE
type,Date,Real
,01, 2.7781436E-02
,02, 4.3615162E-02
,03, 7.5734042E-02
,04, 0.1202449
,05, 0.1635550
,06, 0.1940680
,07, 0.2033970
,08, 0.1888348
,09, 0.1547211
,10, 0.1100197
,11, 6.6857137E-02
,12, 3.6761027E-02
```

1.3.3. PRMS model, 1 parameter, strategy BINARY, timestep ANNUAL_MEAN, period_range 2-7

This step example has a single parameter from the PRMS model and uses the BINARY strategy for calibration. The parameter `psta_nuse` is a distributed parameter with a value of 0 or 1 for each precipitation station in the data file. This parameter is used in the PRMS XYZ precipitation distribution computational method. A value of 1 indicates to include the station in the computation and a value of 0 indicates to exclude the station from the computation. The range indicates that all stations (0-*) will be used in the calibration. The calibration will determine which combination of the selected precipitation stations gives the best fit of the simulated streamflow to the observed streamflow.

The objective function method selected is the absolute value of the difference between the annual mean of the simulated and observed values of streamflow. The period selected is March through August (3-8) so the annual mean will really be the sum of the daily values from March through August divided by the total number of days in the sum.

```
step {
  parameter {
    psta_nuse(lower:0, upper:1, calib_strategy:MEAN)
  }
  objfunc(method:ABSDIF, timestep:ANNUAL_MEAN, period_range:"3-8") {
    sim(file:"out1.csv", table:"yampa-luca", column:"basin_cfs")
    obs(file:"$oms_prj/data/yampaStmBt/yampa_data.csv", table:"obs",
        column:"runoff")
  }
}
```

1.3.4. Monthly Water Balance model, 2 parameters, timestep MONTHLY_MEAN, period_range 1-12

This step has 2 parameters from a monthly water balance model. The time step of the input data to drive the model is monthly. The 2 parameters are distributed parameters used to compute lake outflow as function of storage above lake outflow elevation. In this example the parameters associated with a single lake identified by the range value of 1 will be calibrated.

The objective function method selected is the absolute value of the difference between the monthly mean of the simulated and observed values of lake outflow. The period_range 1-12 indicates that the outflow values for all 12 months will be used to compute the monthly mean over the calibration period. All monthly outflows will be summed and divided by the number of values used to compute the sum to compute the monthly mean.

```
step {
  parameter {
    rout_coef_alpha(lower:1.0, upper:50.0, calib_strategy:MEAN, subset:"1")
    rout_coef_m(lower:1.0, upper:5.0, calib_strategy:MEAN, subset:"1")
  }
  objfunc(method:ABSDIF, timestep:MONTHLY_MEAN, period_range:"1-12") {
    sim(file:"out.csv", table:"Blue-luca", column:"lakesurf_elev[1]")
    obs(file:"out.csv", table:"Blue-luca", column:"runoff[3]")
  }
}
```

```
}
```

1.3.5. Calibration selection example

This step uses the calibration selection to only base calibration on `jh_coef_hru` parameter values for which the `hru_subbasin` is equal to 2. In this example, only `psta_nuse` values with index 1, 4, and 6 (values 14.8, 13.4, and 15.1) would be used for calibration because only the `hru_subbasin` values with those indexes is equal to the range of 1 or 2.

```
Parameter data:
@P, jh_coef_hru,      "{12.6, 14.8, 13.4, 14.7, 13.4, 13.4, 15.1, 12.7}"
@P, hru_subbasin,     "{ 3, 2, 3, 4, 2, 4, 1, 3}"

Luca Code:

...
step {
  parameter {
    jh_coef_hru(lower:0, upper:1, calib_strategy:INDIVIDUAL,
      filter_param:"hru_subbasin", subset:"1,2")
  }
  objfunc(method:ABSDIF, timestep:DAILY, period_range:"1-12") {
    sim(file:"out1.csv", table:"yampa_calibParam", column:"basin_cfs")
    obs(file:"$oms_prj/data/yampaStmBt/yampa_data.csv", table:"obs",
      column:"runoff")
  }
}
...
```

1.4. References

David, O., Markstrom, S.L., Rojas, K.W., Ahuja, L.R., Schneider, I.W. (2002). The Object Modeling System In: Agricultural System Models in Field Research and Technology Transfer, L. Ahuja, L. Ma, T.A. Howell, Eds., Lewis Publishers, CRC Press LLC, 2002: Chapter 15, 317- 331.

David, O., Ascough II, J., Leavesley, G., and L. Ahuja (2010), Rethinking modeling framework design: Object Modeling System 3.0. IEMSS 2010 International Congress on Environmental Modeling and Software – Modeling for Environment’s Sake, Fifth Biennial Meeting, July 5-8, 2010, Ottawa, Canada; Swayne, Yang, Voinov, Rizzoli, and Filatova (Eds.).

Duan, Q., Sorooshian, S. and Gupta, V.K., (1992). Effective and efficient global optimization for conceptual rainfall-runoff models. *Water Resources Research* 28 (4), 1015-1031.

Duan, Q., Sorooshian, S. and Gupta, V.K., (1993). A Shuffled Complex Evolution approach for effective and efficient global minimization. *J. of Optimization Theory and its Applications*, 76 (3), 501-521.

Duan, Q., Sorooshian, S. and Gupta, V.K., (1994). Optimal use of the SCE-UA global optimization method for calibrating watershed models. *Journal of Hydrology*, 158 265-284

Hay, L.E., Umemoto, M., (2006) Multiple-objective stepwise calibration using Luca: U.S. Geological Survey Open-File Report 2006-1323, 25p.

Hay, L.E., Leavesley, G.H., Clark, M.P., Markstrom, S.L., Viger, R.J., and Umemoto, M. (2006). Step-wise, multiple-objective calibration of a hydrological model for a snowmelt-dominated basin. *Journal of the American Water Resources Association*.

Hay, L.E., Leavesley, G.H., and Clark, M.P., (2006). Use of Remotely-Sensed Snow Covered Area in Watershed Model Calibration for the Sprague River, Oregon. Joint 8th Federal Interagency Sedimentation Conference and 3rd Federal Interagency Hydrologic Modeling Conference, Reno, Nevada, April, 2006.

Leavesley, G.H. and L.G. Stannard, (1995). The precipitation-runoff modeling system- PRMS. In: Computer Models of Watershed Hydrology, Water Resources Publications, Highlands Ranch, CO, edited by V.P Singh, Chapter 9, 281-310.

Leavesley, G.H., Restrepo, P.J., Markstrom, S.L., Dixon, M., and Stannard, L.G., (1996). The modular modeling system - MMS: User's manual: U.S. Geological Survey Open File Report 96-151, 200 p.

Index